AD-A071 143 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER VALIDATION SUMMARY REPORT. CONTROL DATA CORPORAT--ETC(U)
JUN 79

UNCLASSIFIED FCCTS/CCVS74-VSR395 NL

1 OF 1
AD A071 143

END
DATE FILMED
8-79
DDC

A066 919

(1) B.S.

LEVEL III

(6) COBOL COMPILER
VALIDATION SUMMARY REPORT

Control Data Corporation
NOS Operating System

(14) FCCTS/
VALIDATION NUMBER CCVS74-VSR395

(11) 19 Jun 79

(12) 94p.

Prepared By:

FEDERAL COMPILER TESTING CENTER
AUTOMATED DATA AND TELECOMMUNICATIONS SERVICE
GENERAL SERVICES ADMINISTRATION
WASHINGTON, D.C. 20406

408 438

UNITED STATES DEPARTMENT OF COMMERCE
National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Date 11 July 1979

NTIS Control # 192093

TO: Defense Documentation Center - DDC/TC
Cameron Station
Alexandria, Virginia 22314

FROM: NTIS, Input Branch
5285 Port Royal Road
Springfield, Virginia 22161

Report # CCVS-74-VSR-395 ADA # ADA071143

Title: "Cobol Compiler - Control Data Corp"

Subject report is ☐ Standard Process ☒ STG report. STG - Special Technology Group. ☐ Computer Product. ☒ Follow up date 19 July 1979

☐ The report will be accessioned by DDC. The form noting the ADA number is returned.

☐ The report has been assigned the ADA number noted above and is returned to NTIS for processing.

☐ DDC will not process the report. It is returned to NTIS.

Mag Tape Price _____

PC & MF Price 10.50 MF 10.50 Source $ ADPES

Stock Quantity 10 Source Share _____

Comments: Bin-778

_____
Signature (for billing)

Copy when completed to
Finance Branch.

DOD Report Action Request
(Replaces NTIS-164 5-72)

Dottie Adams
processor

-A-

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. CCVS74-VSR395	2.	3. Recipient's Accession No.

4. Title and Subtitle

Validation Summary Report # CCVS74-VSR395 (Assigned by Manager of Testing)
CDC NOS COBOL 5.2

5. Report Date: June 19, 1979

6.

7. Author(s)

Same as organization - see 9.

8. Performing Organization Rept. No.

9. Performing Organization Name and Address

Federal Compiler Testing Center (CFT)
Automated Data & Telecommunications Service
General Services Administration
Washington, D.C. 20406

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No.
(C)
(G)

12. Sponsoring Organization Name and Address

Automated Data & Telecommunications Service
General Services Administration
Washington, D.C. 20406

13. Type of Report & Period Covered

14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

This Validation Summary Report (VSR) for the CDC NOS COBOL Compiler Version 5.2/485 (NOS Version 1.3/485) provides a consolidated summary of the results obtained from the validation of the subject compiler against the 1974 COBOL Standard (X3.23-1974/FIPS PUB 21-1). The compiler was validated at the HIGH level of FIPS PUB 21-1. The VSR is made up of several sections showing the discrepancies found. These include an overview of the validation which lists all categories of discrepancies be level/module within X3.23-1974, a section relating the categories of discrepancies to each of the Federal levels of the language; and a detailed listing of discrepancies together with the tests which were failed.

17. Document Analysis a. Descriptors

Programming Languages
Standards
Compilers
COBOL
Verifying
Proving Program Correctness
Software Engineering

b. Identifiers/Open-Ended Terms

CCVS
CVS

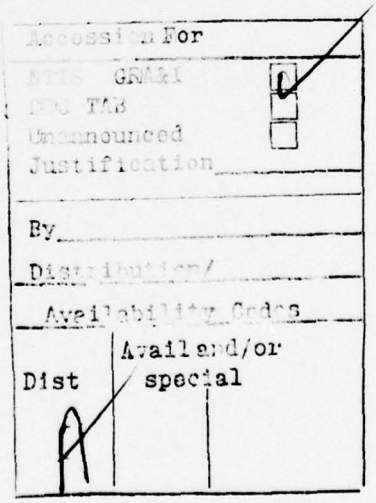
c. COSATI Field/Group 09/02

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited UNCLASSIFIED	UNCLAS	92
	20. Security Class (This Page) UNCLAS	22. Price E06

(See ANSI-Z39.18) See Instructions on Reverse OPTIONAL FORM 272 (4-77) (Formerly NTIS-35) Department of Commerce

79 07 13 058

TABLE OF CONTENTS



COBOL COMPILER VALIDATION

1.	Validation Number	CCVS74-VSR395
2.	Vendor	Control Data Corporation
3.	Mainframe	CYBER 173, SN614
4.	Compiler Identification	COBOL 5.2, Release Level 485 (Product No. F521-46)
5.	Operating System Identification	NOS 1.3, Release Level 485
6.	Other Hardware/Software Environments (*)	

A. CDC 6000 Series
B. CDC CYBER 70 Series except CYBER 76 computer
C. CDC CYBER 170 Series except CYBER 176 computer

7.	Compiler Validation System Version Number	CCVS74 3.0
8.	Federal Information Processing Standard Publication	21-1
9.	Federal COBOL Level Validated	High
10.	Report Date	79/05/01

* List of the additional hardware/software environments on which the COBOL Compiler Validation System was not run but for which the vendor certifies that the identified compiler will produce the same results as noted in this report.

PLEASE NOTE. The Federal Compiler Testing Center may make full and free public disclosure of the Validation Summary Report (VSR) in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of this validation are only for the purpose of satisfying United States Government requirements, and apply only to the Computer System, Operating System release, and compiler version identified in the VSR. The COBOL Compiler Validation System is used to determine, insofar as is practical, the degree to which the subject compiler conforms to the Federal COBOL Standard. Thus, the VSR is necessarily discretionary and judgmental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in the VSR are accurate or complete. The VSR is not meant to be used for the purpose of publicizing the findings summarized therein.

For information concerning this compiler you can contact the vendor's designated representative named below:

Mr. Robert Moylan
Control Data Corporation
6003 Executive Boulevard
Rockville, Maryland 20852

(301)468-8428

1. SECTION 1. INTRODUCTION

1.1 Purpose of the Validation Summary Report

The purpose of the Validation Summary Report (VSR) is to identify individual COBOL language elements whose implementation does not conform to American National Standard Programming Language COBOL, X3.23-1974, and to Federal Standard COBOL as adopted from the American National Standard by Federal Information Processing Standard 21-1 (FIPS PUB 21-1).

1.2 Preparation of the VSR

The Validation Summary Report is prepared by analyzing the results of running the COBOL Compiler Validation System (CCVS). The COBOL Compiler Validation System consists of audit routines containing features of Federal Standard COBOL, their related data, and an executive routine (VP-routine) which prepares the audit routines for compilation. Each audit routine is a COBOL program which includes many tests and supporting procedures indicating the result of the tests.

The testing of a compiler in a particular hardware/operating system environment is accomplished by compiling and executing each audit routine. The report produced by each routine tells whether the compiler passed or failed the tests in the routine. If the compiler rejects some language elements by terminating compilation, giving fatal diagnostic messages, or terminating execution abnormally, then the test containing the code the compiler was unable to process is deleted and the audit routine compilation and execution repeated.

The compilation listings and the output reports of the audit routines constitute the raw data from which the members of the Federal Compiler Testing Center produce a Validation Summary Report.

1.3 Organization of the VSR

The Validation Summary Report is made up of several sections the contents of which are described below.

a. Section 2 summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System. Section 2 is subdivided into a subsection representing each level of each module defined in American National Standard Programming Language COBOL, X3.23-1974. Each subsection contains a list of all of the language elements which must be implemented in order to claim support of that level/module. The list of language elements will be annotated to include a description of both syntax and semantic errors detected during the validation.

b. Section 3 - FIPS PUB 21-1 defines four Federal levels of the COBOL Standard. Section 3.1 of the VSR lists the discrepancies described in Section 2

by the Federal level in which the problem occurs. Section 3.2 lists discrepancies with regard to the Flagging requirement defined in Federal Standard COBOL (the FIPS FLAGGER). Section 3.3 lists discrepancies for the Report Writer Module, which is not a part of Federal Standard COBOL but is contained in X3.23-1974.

c. Section 4 contains information which describes the software environment in which the compiler was tested. This includes the name and version of the operating system; the implementor-names which were used in the Environment Division of the programs comprising the CCVS; the options used with the compiler; and if applicable, information regarding the use of compiler optimization features.

d. Section 5 contains the results of the ASCII validation. The purpose of these tests is to ascertain whether magnetic tapes written in ASCII code and with ANSI standard labels, and card decks with ASCII code, can be transported between the system being validated and a foreign computer system.

e. Appendix A is the Validation Summary Working Document, a working paper resulting from the compilation and execution of the CCVS, and from which the VSR is derived. Not all VSR's will contain this appendix.

1.4 Abstract Covering Compliance to ANS COBOL

Definition of an Implementation of American National Standard Programming Language COBOL (excerpts from X3.23-1974, Chapter 1, Section 1.5).

An implementation is defined to meet the requirements of the American National Standard COBOL specification if that implementation includes a fully implemented specified level of each of the functional processing modules and of the Nucleus as defined in this Standard. It follows from this that, in order to meet the requirements of this Standard, an implementation must:

a. Not require the inclusion of substitute or additional language elements in the source program, in order to accomplish any part of the function of any of the standard language elements.

b. Accept all standard language elements contained in a given level of a module which is specified as being included in the implementation, except as specifically exempted (as pertaining to specific hardware components for which support is not claimed). See "Elements that Pertain to Specific Hardware Components" below.

These points are of particular pertinence in two areas:

(1) There are throughout the American National Standard COBOL specification certain language elements whose syntax, or effect, is specified to be, in part, implementor-defined. While the implementor specifies the con-

straints on that portion of each element's syntax or rules that is indicated cated in this Standard to be implementor-defined, such constraints may not include any requirement for the inclusion in the source program of substitute or additional language elements.

(2) When a function is provided outside the source program that accomplishes a function specified by any particular standard COBOL element, then the implementation must not require, except for Environment Division elements, the specification of that external function in place of or in addition to that standard language element:

The following qualifications apply to the American National Standard COBOL specification:

a. There are certain language elements which pertain to specific types of hardware components. In order for an implementation to meet the requirements of this standard, the implementor must specify the minimum hardware configuration required for that implementation and the hardware components that it supports. Further, when support is thus claimed for a specific hardware component, all standard language elements that pertain to that component must be implemented if the module in which they appear is included in the implementation. Language elements that pertain to specific hardware components for which support is not claimed, need not be implemented. However, the absence of such elements from an implementation of American National Standard COBOL must be specified.

b. An implementation of American National Standard COBOL may include the ENTER statement or not, at the option of the implementor.

c. An implementation that includes, in addition to a specified level of each of the functional processing modules and of the Nucleus, elements or functions that either are not defined in the American National Standard COBOL specification or are defined in a given level of a standard module not otherwise included in the implementation, meets the requirements of this Standard. This is true even though it may imply the extension of the list of reserved words by the implementor, and prevent proper compilation of some programs that meet the requirements of this Standard. The implementor must specify any optional language (language not defined in a specified level but defined elsewhere in the Standard) or extensions (language elements or functions not defined in this Standard) that are included in the implementation.

d. In general, the American National Standard COBOL specification specifies no upper limit on such things as the number of statements in a program, the number of operands permitted in certain statements, etc. It is recognized that these limits will vary from one implementation of American National Standard COBOL to another and may prevent the proper compilation of some programs that meet the requirements of this standard.

IMPLEMENTOR-DEFINED LANGUAGE SPECIFICATIONS

The language elements in the following lists depend on implementor definitions to complete the specificaton of the syntax or rules for the elements.

The elements whose syntax is partly implementor-defined are:

Element	Implementor-Defined Aspect
SOURCE-COMPUTER paragraph	computer-name
OBJECT-COMPUTER paragraph	computer-name
MEMORY SIZE clause	integer
alphabet-name	implementor-name; whether implementor-names are provided.
SPECIAL-NAMES paragraph	implementor-name
ASSIGN clause	implementor-name
VALUE OF clause	implementor-name; whether implementor-names are provided.
RERUN clause	implementor-name and the form; the implementor provides at least one of seven specified forms.
CALL and CANCEL statements	relationship between operand and the referenced program.
COPY statement	relationship between library-name textname, and the library.
ENTER statement	language-name
Margin R	The location.
Area B	The number of character positions.
Qualification	The number of qualifiers; at least five must be supported.

The elements whose effect is partly implementor-defined are:

Element	Implementor-Defined Aspect
alphabet-name	The correspondence between native and foreign character sets.
implementor-name switches	Whether setting can change during execution.
USAGE IS COMPUTATIONAL clause	Representation and whether automatic alignment occurs.
USAGE IS INDEX clause	Representation and whether automatic alignment occurs.
SYNCHRONIZED clause	Whether implicit FILLER positions are generated; their effect on the size of group items and redefining items.
ACCEPT statement	Maximum size of one transfer of data in Level 1 Nucleus.
DISPLAY statement	Maximum size of one transfer of data in Level 1 Nucleus.
Numeric test	Representation of valid sign in the absence of the SIGN IS SEPARATE clause.
Comparison of nonnumeric items	Collating sequence, where NATIVE or implementor-name collating sequence is implicitly or explicitly specified.
Arithmetic expressions	Number of places carried for intermediate results.

Elements That Pertain to Specific Hardware Components

The standard language elements in the list that follows pertain to specific types of hardware components. These language elements must be implemented in an implementation of American National Standard COBOL when support is claimed, by the implementor, for the specific types of hardware components to which they pertain, and the module in which they are defined is included in that implementation.

Element	Hardware Component
CODE-SET clause	Device capable of supporting the specified code.
MULTIPLE FILE TAPE clause	Reel
CLOSE...REEL/UNIT statement	Reel or mass storage
CLOSE...NO REWIND statement	Reel or mass storage
OPEN...REVERSED statement	Reel with the capability of making records available in the reversed order; mass-storage with the capability of making records available in the reversed order.
OPEN...NO REWIND statement	Reel or mass storage
OPEN...I-O statement (Sequential I-O only)	Mass storage
OPEN EXTEND statement	Reel or mass storage
REWRITE statement (Sequential I-O only)	Mass storage
SEND...BEFORE/AFTER ADVANCING statement	Devices capable of vertical positioning; devices capable of action based on mnemonic-names.
USE...I-O (Sequential I-O only)	Mass storage
WRITE...BEFORE/AFTER ADVANCING	Devices capable of vertical positioning; devices capable of action based on mnemonic-name.

1.5 The Federal COBOL Standard

The COBOL compiler validation results enclosed in this document reflect the degree to which the subject COBOL compiler implements the Federal COBOL Standard. The Federal COBOL Standard is essentially the same as the American National Standard Programming Language COBOL, X3.23-1974, with two exceptions:

> The Federal COBOL Standard defines 4 levels and the ANSI Standard defines only the minimum COBOL implementation and the full standard. Low and High levels of the Federal COBOL Standard (see 1.5.1) correspond to the above two ANSI levels (minus the Report Writer module). Two additional levels, low-intermediate and high-intermediate have been included in the Federal Standard between the highest and lowest subsets. These additional levels accommodate hardware which cannot support the full standard, but which is capable of implementing more than the minimum standard.
>
> The Federal COBOL Standard states that the Report Writer Module is not mandatory in any Federal level, but that the specifications contained in X3.23-1974 should be used to the extent practical, consistent with requirements.

The Federal COBOL Standard requires that a compiler contain as a minimum the elements specified in at least one of the Federal levels. No restrictions are imposed on the inclusion of selected features from higher levels or even unique vendor extensions. Compatibility amoung various implementations of a given level containing additional features must be controlled by management imposed standards and restrictions.

1.5.1 Federal Standard COBOL Levels

a. Federal Standard COBOL specifications are the language specifications contained in American National Standard Programming Language COBOL, X3.23-1974. For purposes of the Federal Standard, the modules defined in X3.23-1974 are combined into four levels. Not all computers are large enough to accommodate a COBOL compiler containing the full ANSI Standard. Therefore, the Federal Government requires that all compilers acquired by its agencies contain as a minimum one of the four Federal levels, depending on machine size, configuration and user needs. The knowledge that all computers will support at least one of these four subsets simplifies the task of developing machine-independent COBOL programs.

b. The four levels of Federal Standard COBOL are identified as: Low, Low-Intermediate, High-Intermediate, and High. Each Federal Standard COBOL level is composed of either the high or low levels of the nucleus and ten of the eleven Functional Processing Modules (FPMs) defined in X3.23-1974. The four Federal Standard COBOL levels are reflected in the following table. The numbers in the table refer to the level within the FPM or nucleus as designated in

23-1974, and a dash in the table denotes that the corresponding FPM is omitted.

	Low	Low Intermediate	High Intermediate	High
NUCLEUS	1	1	2	2
FPMs				
TABLE HANDLING	1	1	2	2
SEQUENTIAL I-O	1	1	2	2
RELATIVE I-O	-	1	2	2
INDEXED I-O	-	-	-	2
SORT-MERGE	-	-	1	2
REPORT WRITER	-	-	-	-
SEGMENTATION	-	1	1	2
LIBRARY	-	1	1	2
DEBUG	-	1	2	2
INTER-PROGRAM COMMUNICATION	-	1	2	2
COMMUNICATION	-	-	2	2

1.5.2 Conformance to Federal Standard COBOL

A compiler implemented in conformance to Federal Standard COBOL must meet at least the following requirements.

a. The implementation must include all of the language elements of at least one of the levels of Federal Standard COBOL.

b. The implementation must meet all of the requirements defined in American National Standard COBOL, X3.23-1974, Section I, paragraph 1.5, Definition of An Implementation of American National Standard COBOL which is provided in section 1.4 of this VSR.

c. The implementation must provide a facility for the user to optionally specify a level of Federal Standard COBOL for monitoring his source program at compile time. The monitoring will be an analysis of the syntax used in a source program against the syntax included in the specified level of Federal Standard COBOL. Any syntax used in the source program that does not conform to that

allowed by the user selected level of Federal Standard COBOL will be diagnosed. The syntax diagnosed as not conforming to the specified level will be identified to the user through a diagnostic message on the source program listing. The diagnostic message will contain, at least: (1) The identification of the source program line number in which the nonconforming syntax occurs, (2) the identification of the level of Federal Standard COBOL that supports the syntax or that the syntax is nonstandard COBOL.

1.6 Use of the VSR

The Federal Compiler Testing Center may make full and free public disclosure of the Validation Summary Report (VSR) in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of the validation are only for the purpose of satisfying United States Government requirements, and apply only to the computer system, operating system release, and compiler version identified in the VSR.

The COBOL Compiler Validation System is used to determine, insofar as is practical, the degree to which the subject compiler conforms to the COBOL Standard. Thus, the VSR is necessarily discretionary and judgmental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in the VSR are accurate or complete. The VSR is not meant to be used for the purpose of publicizing the findings summarized therein.

1.7 Sources of Additional Information

FIPS PUB 21-1 defines the Federal COBOL Language Standard. This publication is available from the Office of ADP Standards Management, National Bureau of Standards, Washington, D. C., 20234.

The detailed COBOL language specifications are given in the publication "American National Standard Programming Language COBOL, X3.23-1974", available from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

An explanation of the COBOL Compiler Validation System is contained in the CCVS User's Guide. This document explains how to run the compiler validation system. The User's Guide and a magnetic tape containing a copy of the CCVS programs are available from the National Technical Information Service, Springfield, Virginia, 22151. (Ordering information can be obtained from the Federal Compiler Testing Center.)

1.8 Requests for Interpretation

Questions regarding this VSR or the CCVS in general should be forwarded to the FCTC. If any problem cannot be adequately resolved through the FCTC, the request for interpretation will be forwarded to the Federal COBOL Interpretation Committee for final resolution.

A brochure describing the validation process including the procedures for requesting a validation and resolution of questions involving interpretation of the current Federal Standard is available from the Federal Compiler Testing Center, Automated Data and Telecommunications Service (CFT), General Services Administration, Washington, DC 20406.

1.9 Modules and Language Elements Excluded from Testing

During an official validation, certain CCVS tests may not be used, and certain facilities provided by the subject compiler may not be tested.

1.9.1 Federal Standard COBOL Approved Interpretations

The National Bureau of Standards published in the Federal Register Vol. 41 No. 179, September 14, 1976, an approved interpretation of Federal Standard COBOL as pertains to the evaluation of arithmetic expressions in the COMPUTE statements. This interpretation states that "size of the intermediate result field is implementor-defined."

Since the results of evaluating arithmetic expressions are not predictable, all COMPUTE statements and IF statements containing arithmetic expressions have been removed from the COBOL Compiler Validation System.

1.9.2 Report Writer Module

FIPS PUB 21-1 excludes the Report Writer Module from the Federal COBOL Standard. However, the Report Writer Module is still tested during a validation if support for that module is claimed by the compiler vendor.

1.9.3 Communication Module

Although it is part of Federal Standard COBOL as defined by FIPS PUB 21-1, the Communication Module is not currently tested in the course of an official validation for two specific reasons. First, a large volume of requests for interpretation on this module have been submitted to the cognizant ANSI committee (X3J4) for resolution. Secondly, facilities for testing were insufficient to determine the validity of the Communication Module test programs during the development of CCVS74.

1.9.4 Vendor Omissions or Extensions

Language elements are not tested which have been legitimately omitted from the implementation by the implementor (refer to 1.4). Additionally, no implementor extensions to the standard COBOL language are tested in any way.

1.10 Timeliness of the Validation Summary Reports

The timeliness of the Validation Summary Report is important. Compilers and their related operating system software are modified several times a year. The Compiler Validation System used to validate compilers is also updated during the life of the system. Therefore to ensure that the latest version of both the vendor's compiler and the Validation System are the latest officially released versions, check with the:

Director
Federal Compiler Testing Center
Automated Data and Telecommunications Service (CFT)
General Services Administration
Washington, DC 20406
(703) 557-7806

Please use the Validation Summary Report number of this report when corresponding with the Testing Center.

2. SECTION 2. DETAILED EVALUATION OF ERRORS.

This section summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System (CCVS). The version of the CCVS used during this validation is shown inside the front cover of the VSR.

Section 2 is made up of a variable number of subsections. The number of subsections is dependent on the Level of Federal COBOL being validated. There will be a subsection for each level of each module which is validated. If the high level of a module is validated then there will be two subsections for that module; one for the low level and one for the high level.

A validation of the low level of Federal Standard COBOL would result in three subsections being present. One for Nucleus level 1, one for Sequential I-O level 1, and one for Table Handling level 1.

Each error or deviation noted in this section makes reference to a program or functional COBOL module. If additional information is required or is available it will be contained in the optional Appendix A. If no further information is necessary then Appendix A will not be present.

Each program in the COBOL Compiler Validation System is identified by a 5-character program name. The name associates the routine with the functional processing module and level of American National Standard Programming Language COBOL tested within the program.

For the audit routines which are not flagging routines, a five character name is used which has the general format XXNMM. The first two characters are alphabetic and identify the functional module tested by the program. The permissable values are:

NC - Nucleus
TH - Table Handling
SQ - Sequential I-O
RL - Relative I-O
IX - Indexed I-O
ST - Sort-Merge
RW - Report Writer
SG - Segmentation
LB - Library
DB - Debug
IC - Inter-Program Communication
CM - Communication

The third character of the audit routine name is either a 1 or 2, and identifies the level of the functional module being tested. Each module and level is represented by several programs. The fourth and fifth characters of the program

name are sequence numbers for programs which test features in the same level of the same functional processing module.

As an example, the program name NC210 is the tenth program in the series of routines which test the second level of the Nucleus module.

For the audit routines which are used to validate the flagging requirement of Federal Standard COBOL (1.5.2.c), a five character program name is used which has the format XXYZN. The first two characters XX identify the functional COBOL module being tested by the program and the permissible values are the same as noted above. The third character Y is the character 4 and identifies the program as one of a series which is used for testing compiler flagging. The fourth character Z is the character 2, 3 or 4 and identifies the highest Federal level of the COBOL language needed to support all language elements used in the program. The characters 2, 3 and 4 represent the Low-Intermediate, High-Intermediate and High Levels respectively of Federal Standard COBOL. The fifth character N is a number 1 through 9 which gives a unique name to each program containing features within the same functional processing module and Federal COBOL level.

As an example, the program name LB421 is the first program of a set and the COBOL elements used in the program are available at the Low-Intermediate and higher levels of Federal Standard COBOL. Also, the program is used in testing compiler flagging for the Library Module.

Description of Section 2.

Each error/deviation is noted by number in the left hand margin opposite the language element in question. This number is used in section 3 to categorize errors by Federal level (See 1.5.1). Inserted directly below the language element is a brief description of the error. To the right of the language element is a page reference to X3.23-1974, American National Standard Programming Language COBOL. The reference enclosed in parenthesis at the end of the description is to the appropriate program contained in the Compiler Validation System which gives an example of the of the error/deviation detected. If the suffix to the program name is zeros (e.g., NC000) then the error was detected in the supporting documentation provided with the compiler.

If the optional Appendix A is present, then the program name associated with each error may be suffixed with additional information. For example an A or B following the name represents a syntax or semantic error respectively. A number following the A or B represent the error number which can be found in appendix A.

Example:

```
  2.1 Nucleus Level 1


      .
      .
      .
        Operational symbols: S V P                        II-21
2.1.9       ------------------------------------------------
            * The Picture character 'P' is not supported
            * in this implementaion.
            *                                    (NC101.A.2)
            ------------------------------------------------

      .
      .
      .

  2.2 Sequential I-O Level 1
```

2.1.9 represents the ninth error for Nucleus Level 1

II-21 represents the page in X3.23-1974 where the language element is defined

* Boxes the description of the error/deviation

NC101.A.2 represents:

```
Program name  -  NC101
Syntax error  -  A       (If Appendix A is present)
second error  -  2       (If Appendix A is present)
```

2.1 Nucleus Level 1

2.2 Nucleus Level 2

All elements of 1 NUC 1.2 are a part of 2 NUC 1.2

Language Concepts I-75
  Characters used for punctuation I-65
    . comma
    ; semicolon
  Characters used for arithmetic operations I-52
    + addition
    - subtraction
    * multiplication
    / division
    ** exponentiation
  Characters used in relations I-66
    = equal to
    > greater than
    < less than
  Separators I-75
    The separators, semicolon and comma, are allowed . . II-1
  Character-strings I-76
    COBOL words I-76
      User-defined words I-76
        condition-name
        data-name
          Need not begin with an alphabetic character . . II-1
          May be qualified if necessary for uniqueness . II-1
      Reserved words I-79
        Figurative constants I-80
          ZEROS: ZEROES
          SPACES
          HIGH-VALUES
          LOW-VALUES
          QUOTES
          ALL literal
        Connectives I-79
          Qualifier connectives: OF, IN
          Series connectives: , (separator comma)
              and ; (separator semicolon)
          Logical connectives: AND, OR, AND NOT, OR NOT
  Qualification I-87

Reference format I-105
  Continuation of lines (continuation of words and
      numeric literals is allowed) II-1

Identification Division I-94
  The DATE-COMPILED paragraph II-4

2.3 Table Handling Level 1

2.4 Table Handling Level 2

All elements of 1 TBL 1.2 are a part of 2 TBL 1.2

Data Division
The OCCURS clause III-2
integer-1 TO integer-2 DEPENDING ON data-name

2.4.1 ERROR: OCCURS DEPENDING ON clause

```
**************************************************************************
*       The READ INTO, RETURN FROM and the MOVE statement did not        *
*       function correctly when the receiving data item was described    *
*       with an OCCURS DEPENDING ON clause.                              *
*                                        (SQ219.B, ST216.B and TH221.B)  *
**************************************************************************
```

ASCENDING/DESCENDING data-name
data-name series
ASCENDING/DESCENDING series

Procedure Division
The SEARCH statement III-7
VARYING phrase
AT END phrase
WHEN phrase
The SEARCH ALL statement III-7
AT END phrase
WHEN phrase

2.5 Sequential I-O Level 1

2.6 Sequential I-O Level 2

All elements of 1 SEQ 1.2 are a part of 2 SEQ 1.2

Language Concepts
- Special register I-80
  - LINAGE-COUNTER. IV-3

Environment Division
- The FILE-CONTROL paragraph IV-4
- The file control entry IV-4
  - SELECT clause
    - OPTIONAL phrase
  - RESERVE integer AREA(S) clause
- The I-O-CONTROL paragraph. IV-6
  - SAME RECORD AREA clause
  - SAME RECORD AREA series
  - MULTIPLE FILE TAPE clause

Data Division
- The file description entry IV-10
- The BLOCK CONTAINS clause IV-11
  - integer-1 TO integer-2 RECORDS
  - integer-1 TO integer-2 CHARACTERS
- The LINAGE clause IV-15
  - FOOTING phrase
  - TOP phrase
  - BOTTOM phrase
- The VALUE OF clause IV-19
  - implementor-name IS data-name
  - implementor-name IS data-name series

Procedure Division
- The CLOSE statement IV-20
  - NO REWIND. REMOVAL. or LOCK
  - file-name series
- The OPEN statement. IV-24
  - INPUT
    - REVERSED
    - NO REWIND
  - OUTPUT
    - NO REWIND
  - EXTEND
  - file-name series
  - INPUT. OUTPUT, I-O. and EXTEND series
- The USE statement IV-32
  - EXCEPTION/ERROR PROCEDURE ON file-name series
  - EXCEPTION/ERROR PROCEDURE ON EXTEND

2.6.1 ERROR: WRITE ADVANCING Statement

```
**********************************************************************
*      For the WRITE ADVANCING statement additional line spacing     *
*      occurred when line printing crossed the physical listing page *
*      (page perforation).                                           *
*                                                    (SQ201.B)       *
**********************************************************************
```

```
       BEFORE/AFTER identifier LINES
       BEFORE/AFTER mnemonic-name
       AT END-OF-PAGE imperative-statement
```

2.7 Relative I-O Level 1

2.8 Relative I-O Level 2

All elements of 1 REL 0.2 are a part of 2 REL 0.2

Environment Division
- The FILE-CONTROL paragraph V-5
- The file control entry V-5
  - SELECT clause
  - RESERVE integer AREA(S) clause
  - ACCESS MODE IS DYNAMIC clause
- The I-O-CONTROL paragraph V-7
  - SAME RECORD AREA
  - SAME RECORD AREA entries

Data Division
- The file description entry V-11
- The BLOCK CONTAINS clause V-12
  - integer-1 TO integer-2 RECORDS
  - integer-1 TO integer-2 CHARACTERS
- The VALUE OF clause V-16
  - implementor-name IS data-name
  - implementor-name IS data-name entries

Procedure Division
- The READ statement V-23
  - NEXT RECORD
- The START statement V-28
  - KEY IS phrase
  - INVALID KEY phrase
- The USE statement V-30
  - EXCEPTION/ERROR PROCEDURE
    - ON file-name series

2.9 Indexed I-O Level 1

2.10 Indexed I-O Level 2

All elements of 1 INX 0,2 are a part of 2 INX 0,2

Environment Division
- The FILE-CONTROL paragraph VI-5
- The file control entry VI-5
  - SELECT clause
  - RESERVE integer AREA(S) clause
  - ACCESS MODE IS DYNAMIC clause
  - ALTERNATE RECORD KEY clause
    - WITH DUPLICATES phrase
- The I-O-CONTROL paragraph VI-8
  - SAME RECORD clause
  - SAME RECORD AREA series

Data Division
- The file description entry VI-12
- The BLOCK CONTAINS clause VI-13
  - integer-1 TO integer-2 RECORDS
  - integer-1 TO integer-2 CHARACTERS
- The VALUE OF clause VI-17
  - implementor-name IS data-name
  - implementor-name IS data-name series

Procedure Division
- The READ statement VI-24
  - KEY IS phrase
  - NEXT RECORD
- The START statement. VI-30
  - KEY IS phrase
  - INVALID KEY phrase

2.10.1 ERROR: START statement

```
**************************************************************************
*      The START statement incorrectly positioned the record pointer     *
*      in a program which contained a PROGRAM COLLATING SEQUENCE         *
*      clause.  The presence of the PROGRAM COLLATING SEQUENCE clause    *
*      should have no effect on the comparsions for the START            *
*      statement.                                                        *
*                                                  (IX210.B)             *
**************************************************************************
```

- The USE statement. VI-32
  - EXCEPTION/ERROR PROCEDURE
    - ON file-name series

2.11 Sort-Merge Level 1

Language Concepts
- User-defined words I-76
  - file-name

Environment Division
- The FILE-CONTROL paragraph VII-2
- The file control entry VII-2
  - SELECT clause
  - ASSIGN TO implementor-name clause

Data Division
- FILE SECTION VII-5
- The sort-merge file description entry VII-5
- The DATA RECORDS clause VII-6
- The RECORD CONTAINS clause VII-7

Procedure Division
- The RELEASE statement VII-12
  - FROM phrase
- The RETURN statement VII-13
  - INTO phrase
  - AT END phrase
- The SORT statement (only one SORT statement, a STOP RUN statement, and any associated input-output procedures allowed in the nondeclarative portion of a program) VII-14
  - KEY data-name
    - data-name series
  - ASCENDING series
  - DESCENDING series
  - mixed ASCENDING/DESCENDING
  - INPUT PROCEDURE phrase
    - THRU
  - USING phrase
  - OUTPUT PROCEDURE phrase
    - THRU
  - GIVING phrase

2.12 Sort-Merge Level 2

All elements of 1 SRT 0.2 are a part of 2 SRT 0.2

Environment Division
- The FILE-CONTROL paragraph VII-2
- The file control entry VII-2
  - SELECT clause
- The I-O-CONTROL paragraph VII-3
  - SAME RECORD AREA clause
  - SAME SORT/SORT-MERGE AREA clause
  - SAME series

Procedure Division
- The MERGE statement VII-8
  - KEY data-name
    - data-name series
  - ASCENDING series
  - DESCENDING series
  - mixed ASCENDING/DESCENDING
  - COLLATING SEQUENCE phrase
  - USING phrase
  - OUTPUT PROCEDURE phrase
    - THRU
  - GIVING phrase
- The SORT statement (multiple SORT statements are permitted) VII-14
  - COLLATING SEQUENCE phrase
  - USING phrase (multiple file-names)

2.13 Report Writer Level 1

2.14 Segmentation Level 1

Language Concepts
User-defined words I-76
segment-number

Procedure Division
Segment-numbers IX-4
Fixed segment-number range 0 through 49
Non-fixed segment-number range 50 through 99
All sections with the same segment-number must
be together in the source program

2.15 Segmentation Level 2

All elements of 1 SEG 0,2 are a part of 2 SEG 0,2

Environment Division
The OBJECT-COMPUTER paragraph
SEGMENT-LIMIT IX-5

Procedure Division
Segment-numbers IX-4
Sections with the same segment-number need not
be physically contiguous in the source program

2.16 Library Level 1

Language Concepts
User-defined words I-76
text-name

All divisions
The COPY statement X-2

2.17 Library Level 2

All elements of 1 LIB 0,2 are a part of 2 LIB 0,2

Language Concepts
- User-defined words I-76
  - library-name

All divisions
- The COPY statement X-2
  - OF library-name
  - REPLACING phrase

2.18 Debug Level 1

2.19 Debug Level 2

All elements of 1 DEB 0,2 are a part of 2 DEB 0,2

Procedure Division
USE FOR DEBUGGING statement XI-4
ALL REFERENCES OF identifier series
file-name series
cd-name series

2.20 Inter-Program Communication Level 1

2.21 Inter-Program Communication Level 2

All elements of 1 IPC 0.2 are a part of 2 IPC 0.2

Procedure Division
- The CALL statement XII-5
  - identifier
  - ON OVERFLOW phrase
- The CANCEL statement XII-7

2.22 Communication Level 1

---

* The COMMUNICATION Module is not currently evaluated as
* part of an official validation. See Section 1.9.3.

Language Concepts
- User-defined words I-76
  - cd-name

Data Division
- COMMUNICATION SECTION XIII-2
- The communication description entry XIII-3
  - FOR INPUT clause
    - END KEY
    - MESSAGE COUNT
    - MESSAGE DATE
    - MESSAGE TIME
    - SYMBOLIC QUEUE
    - SYMBOLIC SOURCE
    - SYMBOLIC SUB-QUEUE-n
    - STATUS KEY
    - TEXT LENGTH
  - FOR OUTPUT clause
    - DESTINATION COUNT
    - DESTINATION TABLE
      - INDEXED BY
    - ERROR KEY
    - SYMBOLIC DESTINATION
    - STATUS KEY
    - TEXT LENGTH

Procedure Division
- The ACCEPT MESSAGE COUNT statement XIII-12
- The DISABLE statement XIII-13
  - INPUT
  - OUTPUT
  - KEY identifier/literal
- The ENABLE statement XIII-15
  - INPUT
  - OUTPUT
  - KEY identifier/literal
- The RECEIVE statement XIII-17
  - MESSAGE
  - INTO identifier
  - NO DATA phrase
- The SEND statement XIII-20

```
FROM identifier-1 WITH
WITH EMI
WITH EGI
BEFORE/AFTER ADVANCING
   identifier-3 LINES
   integer LINES
   mnemonic-name
   PAGE
```

2.23 Communication Level 2

```
-----------------------------------------------------------------
*    The COMMUNICATION Module is not currently evaluated as
*    part of an official validation.  See Section 1.9.3.
-----------------------------------------------------------------
```

All elements of 1 COM 0.2 are a part of 2 COM 0.2

Communication Section

The communication description entry. XIII-3
FOR INPUT
INITIAL

Procedure Division

The DISABLE statement. XIII-13
INPUT
TERMINAL

The ENABLE statement XIII-15
INPUT
TERMINAL

The RECEIVE statement. XIII-17
SEGMENT

The SEND statement XIII-20
FROM identifier-1
WITH identifier-2
WITH ESI

3. SECTION 3. COMPILER STATUS

3.1 Federal Standard COBOL

Section 1.5 explains the four levels of Federal Standard COBOL and their relation to American National Standard COBOL. This section lists the discrepancies described in Section 2 by the Federal level in which the problem occurs. All errors listed for a lower level are also errors in any higher level, even though they are listed only in the lower level. The paragraph number from Section 2 is used to reference the errors in each Federal level. When the subject compiler fails to support an entire functional module, that error indication appears in upper case to highlight its seriousness in comparison with other errors which may involve only a particular language element or facility.

3.1.1 Low Level

None

3.1.2 Low-Intermediate Level

None

3.1.3 High-Intermediate Level

2.4.1	TH221	OCCURS DEPENDING ON clause in a receiving data item did not function correctly when used with a MOVE, READ . . . INTO, or RETURN . . . INTO statement.
2.6.1	SQ201	WRITE ADVANCING resulted in additional line spacing.

3.1.4 High Level

2.9.1	IX210	START statement did not function correctly when the program contained a COLLATING SEQUENCE clause specifying a sequence other than native.

3.2 Federal Standard COBOL Flagging

A requirement of Federal Standard COBOL is that the COBOL implementation include a facility for flagging COBOL elements which do not conform to a specified level of Federal Standard COBOL. This section lists the flagging discrepancies described in Section 2 by Federal COBOL level.

3.2.1 Low Level

3.2.1.1	IC431	CALL USING data-name series was flagged with an incorrect message which indicated that it was a feature of Low-Intermediate level COBOL.
3.2.1.2	RL421	OPEN file-name series of Relative I-O was not flagged as feature of Low-Intermediate level COBOL.
3.2.1.3	RL421	CLOSE file-name series of Relative I-O was not flagged as a feature of Low-Intermediate level COBOL.
3.2.1.4	RL431	USE file-name series statement of Relative I-O was not flagged as a feature of High-Intermediate level COBOL.
3.2.1.5	SQ431	OPEN file-name series statement of Sequential I-O was not flagged as a feature of High-Intermediate level COBOL.
3.2.1.6	SQ431	CLOSE file-name series statement of Sequential I-O was not flagged as a feature of High-Intermediate level COBOL.
3.2.1.7	SQ431	USE file-name series statement of Sequential I-O was not flagged as a feature of High-Intermediate level COBOL.
3.2.1.8	ST431	SORT statement was flagged with an unnecessary flagging message.

3.2.2 Low-Intermediate Level

3.2.2.1	IC431	CALL USING data-name series was flagged with an incorrect message which indicated that it was a feature of Low-Intermediate level COBOL.
3.2.2.2	NC431	Continuation lines were not flagged.
3.2.2.3	RL421	OPEN file-name series statement of Relative I-O should not be flagged at the Low-Intermediate level COBOL.
3.2.2.4	RL421	CLOSE file-name series statement of Relative I-O should not be flagged at Low-Intermediate level COBOL.
3.2.2.5	RL431	USE file-name series statement of Relative I-O was not flagged as a feature of High-Intermediate level COBOL.
3.2.2.6	SQ431	OPEN file-name series statement of Sequential I-O was not flagged as feature of High-Intermediate level COBOL.
3.2.2.7	SQ431	CLOSE file-name series statement of Sequential I-O was not flagged as a feature of High-Intermediate level COBOL.
3.2.2.8	SQ431	USE file-name series statement of Sequential I-O was not flagged as a feature of High-Intermediate level COBOL.

3.2.2.9 ST431 SORT statement was flagged with an unnecessary flagging message.

3.2.3 High-Intermediate Level

3.2.3.1 RL421 OPEN file-name series statement of Relative I-O should not beflagged at the High-Intermediate level COBOL.

3.2.3.2 RL421 CLOSE file-name series statement of Relative I-O should not be flagged at the High-Intermediate level COBOL.

3.2.3.3 RL431 USE file-name series statement of Relative I-O should not be flagged at the High-Intermediate level COBOL.

3.2.3.4 SQ431 OPEN file-name series statement of Sequential I-O should not be flagged at the High-Intermediate level COBOL.

3.2.3.5 SQ431 CLOSE file-name series statement of Sequential I-O should not be flagged at the High-Intermediate level COBOL.

3.2.3.6 SQ431 USE file-name series statement of Sequential I-O should not be flagged at the High-Intermediate level COBOL.

3.2.3.7 SQ431 WRITE AT END-OF-PAGE statement of Sequential I-O should not be flagged at the High-Intermediate level COBOL.

3.2.3.8 ST431 SORT statement was flagged with an unnecessary flagging message.

3.2.4 High Level

3.2.4.1 SQ431 WRITE AT END-OF-PAGE of Sequential I-O should not be flagged at the High level COBOL.

3.3 American National Standard COBOL

Full American National Standard COBOL consists of the entire set of language elements defined in the ANSI COBOL standard (refer to 1.7). It is also the equivalent of high level Federal Standard COBOL plus the Report Writer module. Therefore, this section lists only those discrepancies found while validating the Report Writer Module.

No errors

4. SECTION 4. SOFTWARE ENVIRONMENT

The compiler referenced in this document was validated using the software environment described in this section. When using a modification of the described environment, the compiler may or may not continue to conform to the Standard. It should be noted that during the validation process, an attempt is made to validate as many different options as possible.

The use of compiler options, implementor-names in the Environment Division and any form of optimization which is not described in this report could cause the compiler to produce a program that does not perform according to the specifications of Standard COBOL. Only the environment described in this document has been used with this compiler to satisfy the requirements of FIPS PUB 21-1 and FPMR 101-32.1305.1a. (Any deviations which must be corrected as per the referenced FPMR are described in Sections 2 and 3 of this report.)

4.1 Compiler options used

Options specified:	B=BINARY1	(disposition of binary output)
	CPY	(compile all COPY statements)
	X=COBOLC1	(name of library file)
	SB	(compile as subprogram)
	PD	(listings printed single space at 8 lines per inch)
	LO=S/M/R	(source listings options - source/ map/cross reference)

(Not all options were used for all programs)

The compiler flagging options are:	FIPS=1	(Low Level)
	FIPS=2	(Low-Intermediate Level)
	FIPS=3	(High-Intermediate Level)
	FIPS=4	(High Level)

Option defaulted:

None

4.2 Environment Division implementor-names

Printer destined files

'OUTPUT' USE 'PRINTF=YES'

Tape Files

For single-file reels:
  TAPEnn (where n is a numeric digit)
For multi-file reels:
  MULTIOn

Sequential I-O Mass-storage files

Single-unit:
  MSDTSCn
Multi-unit:
  UNITnn

Relative I-O Mass-storage files

RELn

Indexed I-O Mass-storage files

ISFILEn INDEXn

Sort files (SD)

SORTFLn

Switch names

SWITCH-1
SWITCH-2

Source Computer name

CYBER

Object Computer name

CYBER

4.3 Optimization

The compiler may or may not have optimization features. If optimization is available by option, it was used during the validation process (during a separate execution of the Compiler Validation System) to determine if its use causes the compiler to produce a program which does not give the expected results. If the optimization is invoked through the compiler call statement then it is mentioned in paragraph 1 above. If it is invoked through the

introduction of syntax in other than the Data and Procedure Divisions of the source program it is shown below. Optimization which would require modification to the Data and Procedure Divisions is not considered in this report in that it is beyond the scope of the use of standard COBOL and the validation process.

This system does not have a selectable optimizaton feature for the compiler.

4.4 Compiler

Control Data Corporation, COBOL 5.2, Release Level 485

4.5 Operating System

Control Data Corporation, NOS 1.3, Release Level 485

4.6 COBOL Reference Manuals

Control Data Corporation
COBOL. Version 5
Reference Manual
Publication No. 60497100

Control Data Corporation
COBOL. Version 5
User's Guide
Publication No. 60497200

5. SECTION 5. ASCII VALIDATION

5.1 Purpose of ASCII Validation

The ASCII Validation is performed by running a sequence of three CCVS74 programs (SQ118. SQ119. SQ120) using special procedures. The purpose of this special run is to validate that the compiler/operating system being tested is capable of processing ASCII code represented on magnetic tape and punched cards that were produced (in accordance with the appropriate American National Standard) by another system. There is also a magnetic tape and a card file created during the validation which will be taken to another system for further processing. The purpose is to determine whether the compiler/operating system being tested can also produce ASCII representation on magnetic tape and punched cards which can be processed by a another computer system.

5.2 Applicable ANSI Standards

The ASCII Validation is based on several American National Standards and presumes their support by the compiler/operating system being validated. These are:

1. American National Standard Programming Language COBOL X3.23-1974.

   - The CODE-SET clause is used to read and write the ASCII files.

   - The PROGRAM COLLATING SEQUENCE clause is used to process the data in ASCII mode as well as native mode.

   - The SIGN...SEPARATE clause is used for signed data and all data is in the DISPLAY (character) mode.

2. American National Standard Code for Information Interchange (ASCII) X3.4-1968. (Note that this describes the code, not the labeling and tape recording formats.)

3. American National Standard Hollerith Punched Card Code. X3.26-1970.

4. American National Standard Magnetic Tape Labels for Information Interchange, X3.27-1969.

5. American National Standard Recorded Magnetic Tape

for Information Interchange (800 CPI, NZRI). X3.22-1967.

6. American National Standard Recorded Magnetic Tape for Information Interchange (1600 CPI, PR). X3.39-1973.

The language of the 1974 COBOL Standard provides the capability to accept, process, and produce ASCII code. The ASCII Standard describes the code insofar as the bit arrangement and configuration, but does not address recording techniques, record formats or any labeling scheme. The 800 CPI, NZRI magnetic tape recording standard was used to establish the recording density and techniques. (1600 CPI, PE based on X3.39-1973 "Recorded Magnetic Tape for Information Interchange" could be used under special arrangements.) The tape labeling scheme used in these tests is based on X3.27-1969 but is also compatible with the revision to that tape label standard. Only the VOL1, HDR1, and EOF1 labels are used. The records are fixed length and unblocked.

5.3 ASCII Validation Process

During the validation, the Validation Manager for the Federal COBOL Compiler Testing Service uses the ASCII-encoded magnetic tape and card files in addition to the normal tape files associated with a validation. For the ASCII portion of the validation the following steps are performed:

1. The tape file and card deck (produced on another computer system) are used as input to several programs designed to validate whether the system being validated can accept and process the data as defined by the respective standards. Any changes made during this validation to the source programs reading the data are noted below in 5.4.1.

2. A tape file and card file are produced during the validation which should prove to be identical to the files described in 1 above. These two files are then processed on a different computer system to determine the degree to which the system being validated supports the ASCII standard. Any changes made during this validation to the source program producing the data are noted below in 5.4.2.

5.4 Results for This Validation

1. The CYBER 73 system was able to successfully process both the ANSI labeled magnetic tape and the Hollerith card deck (both with ASCII code representations). No program modifications were necessary.

2. The CYBER 73 system produced both an ANSI labeled magnetic tape and a Hollerith card deck, which were verified later on a foreign system as being correct in ANSI format and ASCII character set. No program modifications were necessary.

A. APPENDIX A. VALIDATION SUMMARY REPORT WORKING DOCUMENT

This appendix is a working paper produced during the validation and documents the results of the compilation and execution of each of the programs comprising the CCVS. The results contained herein are based on the use of the compiler within the Validation Environment identified in this appendix. This appendix (Validation Summary Working Document) is not part of the official Validation Summary Report (VSR) and is not intended to reflect in any way the compiler's usefulness or degree of conformance to the language specifications.

The reader of this appendix should keep in mind that the same problem area may appear in more than one program, but is considered only as one single discrepancy and as such is reflected only once in the body of the VSR. (The VSR will in turn only reference the first occurrence of the problem in the appendix.)

The reference documents for COBOL are American National Standard Programming Language COBOL (X3.23-1974), and Federal Standard COBOL (FIPS PUB 21-1).

A.1 Validation Environment

COMPILER IDENTIFICATION:	COBOL 5.2, Release level 485 (Product No. F521-46)
COMPUTER SYSTEM:	Control Data Corporation, CYBER 173, SN614 657 and 659 tape drives 844 single and double density disks 131K memory with 20 PPUs
OPERATING SYSTEM:	NOS 1.3, Release Level 485
FIPS COBOL LEVEL VALIDATED:	High

A.2 Run Summaries

COMMUNICATION LEVEL 1 and LEVEL 2

No Communication programs were run. See Section 1.9.3.

COMMUNICATION MODULE FLAGGING

CM431

A. Compilation

1. Low Level Flagging

No errors

2. Low-Intermediate Level Flagging

No errors

3. High-Intermediate Level Flagging

No errors

4. High Level Flagging

No errors

B. Execution

No errors

DEBUG MODULE LEVEL 1

DB101 through DB105

A. Compilation

No errors

B. Execution

No errors

DEBUG MODULE LEVEL 2

DB201 through DB204

A. Compilation

No errors

B. Execution

No errors

DEBUG MODULE FLAGGING

DB421 through DB422

A. Compilation

1. Low Level Flagging

No errors

2. Low-Intermediate Level Flagging

No errors

3. High-Intermediate Level Flagging

No errors

4. High Level Flagging

No errors

B. Execution

No errors

DB431

A. Compilation

1. Low Level Flagging

No errors

2. Low-Intermediate Level Flagging

No errors

3. High-Intermediate Level Flagging

No errors

4. High Level Flagging

No errors

B. Execution

No errors

APPENDIX A

INTER-PROGRAM COMMUNICATION MODULE LEVEL 1

IC101 through IC123

A. Compilation

No errors

B. Execution

No errors

IC151 through IC152

A. Compilation

No errors

B. Execution

No errors

INTER-PROGRAM COMMUNICATION MODULE LEVEL 2

IC201 through IC208

A. Compilation

No errors

B. Execution

No errors

INTER-PROGRAM COMMUNICATION MODULE FLAGGING

IC421 through IC422

A. Compilation

1. Low Level Flagging

   No errors

2. Low-Intermediate Level Flagging

   No errors

3. High-Intermediate Level Flagging

   No errors

4. High Level Flagging

   No errors

B. Execution

   No errors

IC431 through IC432

A. Compilation

1. Low Level Flagging

   The compiler produced a flagging message on the CALL statement

   CALL SUB-PROG-NAME USING DATA-N1: DATA-N2

   indicating it was a feature of Low-Intermediate Level COBOL. (.e., FIPS=2) of Federal Standard COBOL. The CALL identifier USING data-name series is a feature of High-Intermediate Level COBOL (i.e., FIPS=3).

2. Low-Intermediate Level Flagging

   Same as A.1 above.

3. High-Intermediate Level Flagging

   No errors

4. High Level Flagging

   No errors

B. Execution

No errors

INDEXED I-O MODULE LEVEL 1

IX101 through IX107

A. Compilation

No errors

B. Execution

No errors

INDEXED I-O MODULE LEVEL 2

IX201 through IX209

A. Compilation

No errors

B. Execution

No errors

IX210

A. Compilation

No errors

B. Execution

Several READ and START tests failed. This program includes a PROGRAM COLLATING SEQUENCE clause for the purpose of testing the affect of this clause on the START statement. All nonnumeric comparsion rules, as specified by the relational operator of the KEY phrase of the START statement, apply except that the presence of the PROGRAM COLLATING SEQUENCE clause will have no effect on the comparsion. It appears this systems uses the PROGRAM COLLATING SEQUENCE clause in processing Indexed I-O files.

IX211

A. Compilation

No errors

B. Execution

No errors

INDEXED I-O MODULE FLAGGING

IX441

A. Compilation

1. Low Level Flagging

All occurrences of the language elements

OPEN file-name series
CLOSE file-name series

were flagged with the message

30 7819 THE (OPEN or CLOSE) FILE-NAME SERIES IS BEING DIAGNOSED FOR ANY FIPS LEVEL LESS THAN FIPS=4 AND WITHOUT REGARD TO FILE-NAME ORGANIZATION.

The OPEN and CLOSE statement of Indexed I-O is a feature of High Level of Federal Standard COBOL.

2. Low-Intermediate Level Flagging

Same as IX441.A.1 above.

3. High-Intermediate Level Flagging

Same as IX441.A.1 above.

4. High Level Flagging

No errors

B. Execution

No errors

IX442

A. Compilation

1. Low Level Flagging

The same compiler message as noted in IX441.A.1 was produced on the USE file-name series statement in this program.

2. Low-Intermediate Flagging

Same as IX442.A.1 above.

3. High-Intermediate Level Flagging

Same as IX442.A.1 above.

4. High Level Flagging

No errors.

APPENDIX A

LIBRARY MODULE LEVEL 1

LB101 through LB107

A. Compilation

No errors

B. Execution

No errors

LIBRARY MODULE LEVEL 2

LB201 through LB207

A. Compilation

No errors

B. Execution

No errors

LIBRARY MODULE FLAGGING

LB421 through LB441

A. Compilation

1. Low Level Flagging

No errors

2. Low-Intermediate Level Flagging

No errors

3. High-Intermediate Level Flagging

No errors

4. High Level Flagging

No errors

B. Execution

No errors

NC120

A. Compilation

No errors

B. Execution

No errors

NC151 through NC165

A. Compilation

No errors

B. Execution

No errors

NUCLEUS MODULE LEVEL 2

NC201 through NC219

A. Compilation

No errors

B. Execution

No errors

NUCLEUS MODULE FLAGGING

NC431

A. Compilation

1. Low Level Flagging

   The line reference number noted in the flagging message for detecting nested REDEFINES clauses points to the original REDEFINES clause rather than to the nested REDEFINES clause.

2. Low-Intermediate Level Flagging

   a. Same as NC431.A.1 above.

   b. Source lines which are continuation of COBOL words were not flagged.

3. High-Intermediate Level Flagging

   No errors

4. High Level Flagging

   No errors

B. Execution

   No errors

RELATIVE I-O MODULE LEVEL 1

RL101 through RL109

A. Compilation

No errors

B. Execution

No errors

RL151 through RL153

A. Compilation

No errors

B. Execution

No errors

RELATIVE MODULE LEVEL 2

RL201 through RL205

A. Compilation

No errors

B. Execution

No errors

RELATIVE I-O MODULE FLAGGING

RL421

A. Compilation

1. Low Level Flagging

   All occurrences of the OPEN file-name series and the CLOSE file-name series were flagged with the message:

   30 7819 THE (OPEN or CLOSE) FILE-NAME SERIES IS BEING DIAGNOSED FOR ANY FIPS LEVEL LESS THAN FIPS=4 AND WITHOUT REGARD TO FILE-NAME ORGANIZATION.

   The OPEN and CLOSE statements for Relative I-O are a feature of Low-Intermediate Level of Federal Standard COBOL.

2. Low-Intermediate Level Flagging

   Same as RL421.A.1 above.

3. High-Intermediate Level Flagging

   Same as RL421.A.1 above.

4. High Level Flagging

   No errors

RL431

A. Compilation

1. Low Level Flagging

The same flagging message as noted in RL421.A.1 was produced for the USE file-name series statement in this program. The USE file-name series is a feature of High-Intermediate Level of Federal Standard COBOL.

2. Low-Intermediate Level Flagging

   Same as RL431.A.1 above.

3. High-Intermediate Level Flagging

   Same as RL431.A.1 above.

4. High Level Flagging

No errors

B. Execution

No errors

REPORT WRITER MODULE LEVEL 1

RW101 through RW104

A. Compilation

No errors

B. Execution

No errors

SEGMENTATION MODULE LEVEL 1

SG101 through SG106

A. Compilation

No errors

B. Execution

No errors

SEGMENTATION MODULE LEVEL 2

SG201 through SG204

A. Compilation

No errors

B. Execution

No errors

SEGMENTATION MODULE FLAGGING

SG421 and SG441

A. Compilation

1. Low Level Flagging

No errors

2. Low-Intermediate Level Flagging

No errors

3. High-Intermediate Level Flagging

No errors

gh Level Flagging

No errors

B. Execution

No errors

APPENDIX A

SEQUENTIAL I-O MODULE LEVEL 1

SQ101 through SQ121

A. Compilation

No errors

B. Execution

No errors

SQ151 through SQ153

A. Compilation

No errors

B. Execution

No errors

SEQUENTIAL I-O MODULE LEVEL 2

SQ201

A. Compilation

No errors

B. Execution

In test WRT-TEST-09 additional blank lines appeared within the logical page when the line printing crossed onto another physical listing page (crossed page perforation) during execution of a WRITE ADVANCING statement.

SQ202

A. Compilation

No errors

B Execution

Several WRITE ADVANCING statements in this program produced the same results as noted in SQ201.B above.

SQ203 through SQ213

A. Compilation

No errors

B. Execution

No errors

SQ214

A. Compilation

No errors

B. Execution

1. Test WRT-TEST-02 is a test of a WRITE BEFORE ADVANCING containing an EOP phrase. The LINAGE clause value is set to 40 and there is no FOOTING phrase specified. The test expects 40 detailed lines to be printed on the 1st logical page. 20 detailed lines printed on the 2nd page and the EOP line printed on the first line of the 2nd logical page. This system printed 39 detailed lines and the EOP line on the 1st logical page.

2. Test WRT-TEST-03 is a test of a WRITE AFTER ADVANCING statement containing an EOP phrase. The LINAGE clause value is set to 40 and there is no FOOTING phrase specified. The test expects 39 detailed lines to be printed on the 1st logical page. the remaining detailed lines printed on the 2nd page and the EOP line to follow detailed line 40 and be the 2nd line of the 2nd logical page. This system printed the correct number of detailed lines per respective logical page but printed the EOP line as the first line on the 2nd logical page (before detailed line 40) instead of the 2nd line.

3. Several WRITE ADVANCING statements in this program produced the same results as noted in SQ201 above.

SQ215

A. Compilation

No errors

B. Execution

Same as SQ214.B.3 above.

SQ216 through SQ218

A. Compilation

No errors

B. Execution

No errors

SQ219

A. Compilation

No errors

B. Execution

Test READ-TEST-3 reads a record (using the READ INTO statement) into a group item which includes a data item with an OCCURS DEPENDING ON clause. Before the operation is performed the ODO value is set to 5 (maximum possible value is 9). the read is executed. the ODO value is reset to 9 and then the group item is checked. Only the number of occurances active (5) at the time the INTO is executed should participate in the operation.

Computed result: "123456789"
Expected result: "12345xxxx"

Where 'xxxxx' should not be '6789' unless the maximum length (9) was used in the operation instead of the active length (5).

SQ220

A. Compilation

No errors

B. Execution

No errors

SEQUENTIAL I-O MODULE FLAGGING

SQ431

A. Compilation

1. Low Level Flagging

The following Language elements

```
USE ... file-name series
OPEN OUTPUT  file-name series
CLOSE file-name series
OPEN INPUT file-name-1 OUTPUT file-name-2
```

were flagged with the message

```
30 7819  THE (USE or OPEN or CLOSE) FILE-NAME SERIES IS BEING
         DIAGNOSED FOR ANY FIPS LEVEL LESS THAN FIPS=4 AND
         WITHOUT REGARD TO FILE-NAME ORGANIZATION.
```

The file-name series of the above statements for Sequential I-O is a feature of High-Intermediate Level and higher levels of Federal Standard COBOL.

2. Low-Intermediate Level Flagging

Same as SQ431.A.1 above.

3. High-Intermediate Level Flagging

a. Same as SQ431.A.1 above.

b. The flagging message

```
19 7826  FIPS=3  SUPPORTS AT END-OF-PAGE
                 IMPERATIVE-STATEMENT
```

was issued on a WRITE AT END-OF-PAGE statement. There should not be any flagging message produced on this elements at this level

of compiler monitoring.

4. High Level Flagging

Same as SQ431.A.3.b above.

B. Execution

No errors

SORT MODULE LEVEL 1

ST101 through ST118

A. Compilation

No errors

B. Execution

No errors

SORT MODULE LEVEL 2

ST201

A. Compilation

No errors

B. Execution

No errors

ST216

A. Compilation

No errors

B. Execution

Tests RETURN-TEST-1 and RETURN-TEST-2 failed. These tests use the RETURN INTO identifier statement to return a record from the sort into an identifier which is described with an OCCURS DEPENDING ON clause. Prior to execution of the RETURN statement the ODO is set to a value less than its maximum number of occurrences. This system uses the maximum permitted size of the identifier for the RETURN INTO operation regardless of the ODO value. (See SQ219 or TH221 for additional information regarding the problem which caused this test to fail.)

ST217 through ST218

APPENDIX A

A. Compilation

No errors

B. Execution

No errors

SORT MODULE FLAGGING

ST431 thru ST434

A. Compilation

1. Low Level Flagging

For flagging multiple sorts the following message appears regardless of whether there is one or more than one SORT statements in the program:

PLEASE VERIFY THAT YOUR PROCEDURE DIVISION CONTAINS ONLY THE BASIC SORT PROGRAM ALLOWED BY LEVEL 1 OF THE SORT MODULE

2. Low-Intermediate Level Flagging

Same as A.1 above.

3. High-Intermediate Level Flagging

Same as A.1 above.

4. High Level flagging

No errors

B. Execution

No errors

ST441 thru ST443

A. Compilation

1. Low Level Flagging

Same as ST431.A.1 above.

2. Low-Intermediate Level Flagging

Same as ST431.A.1 above.

3. High-Intermediate Level Flagging

Same as ST431.A.1 above.

4. High Level Flagging

No errors

B. Execution

No errors

TABLE HANDLING LEVEL 1

TH101 through TH111 and TH151 through TH152

A. Compilation

No errors

B. Execution

No errors

TABLE HANDLING LEVEL 2

TH201 through TH220

A. Compilation

No errors

B. Execution

No errors

TH221

A. Compilation

No errors

B. Execution

Test MOVE-TEST-4 did not produce the expected result. This test sets the ODO value for the recieving field to 9. moves the value 'PASS' into positions 6 through 9 of the variable length data item and resets the ODO value to 5 - indicating that only 5 of the 9 occurances of the variable length data item are active. Then the value '9 ACTIVE: TEST FAIL' is moved to the group containing the variable length data item. The characters 'FAIL' should not be moved since they are beyond the active portion of the variable length data item. The ODO value is once again set to 9 and the positions 6 through 9 are examined to see if the data corresponds to the value moved which was beyond the active area of the variable length data item.

Computed Result: "9 ACTIVE: TEST FAIL"
Expected Result: "9 ACTIVE: TEST xxxx"

Where 'xxxx' would not contain 'FAIL' unless the maximum length (9) was used in the operation instead of the active length (5).

TABLE HANDLING MODULE FLAGGING

TH431

1. Low Level Flagging

   No errors

2. Low-Intermediate Flagging

   No errors

3. High-Intermediate Flagging

   No errors

4. High Level Flagging

   No errors

B. Execution

No errors